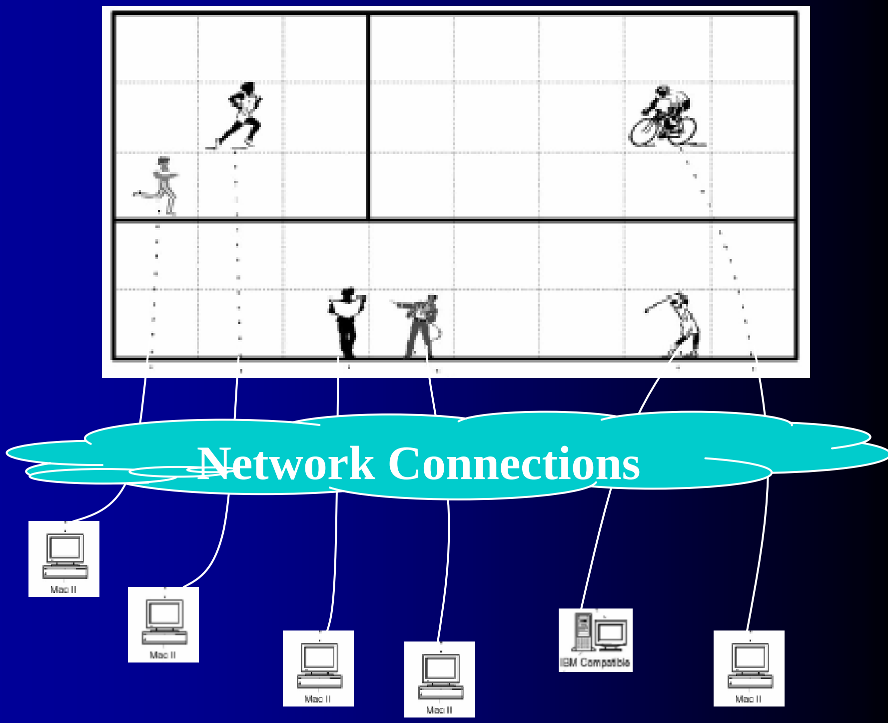
Gamelet: A Mobile Service Component for Building Multi-server DVE on Grid

Tianqi Wang, Cho-Li Wang, Francis C.M.Lau
Department of Computer Science
The University of Hong Kong

What is DVE ?

A Distributed Virtual Environment (DVE) system is a software system through which people who are geographically dispersed over the world can interact with each other by sharing a consistent environment in terms of space, presence and time.



- realistic 3D graphics
- real-time interaction
- a large number of users

Many DVE Applications

- Military team training
- Collaborative design and engineering
- Increasingly used for:
  - virtual shopping mall
  - Interactive e-learning
  - Multiplayer online games

General Client-server Model

Server Tasks:

- Receive user messages
- Calculate world state
- Ensure objects consistency
- Message filtering/compressing
- Administration work

Client Tasks:

- Scene rendering
- Simple calculations

What are the problems

- World State Consistency
- Real-time Response

Grid Technology

- The number of users is unpredictable.
  - Support scalability and dynamic resource aggregation
- Workload imbalance among servers as users may act freely in the virtual world
  - Need dynamic load transfer support

Grid Computing

- Grid Computing (I. Foster)
  - Concerned with flexible, secure, coordinated resource sharing among dynamic collections of virtual organizations
- Grid Service:
  - A kind of stateful, transient web service
  - Large-scale scientific experiments (DOE Science Grid)
  - Large-scale data analysis (EU Data Grid)

DVE Systems on Grid

- Butterfly Grid
  - Easy to use **commercial** grid computing environment for developers
  - High-performance networked servers for the publishers
- Cal-(IT)^2 Game Grid
  - Provides the first massively multi-user online game grid for research, teaching, art, and experimentation

Challenges

- How to re-design the existing DVE system into an <u>open and service oriented</u> system
  - Map monopolistic model into OGSA framework
- How to ensure the <u>quality of service</u> from the end users' perspective
  - Dynamic load migration/balancing

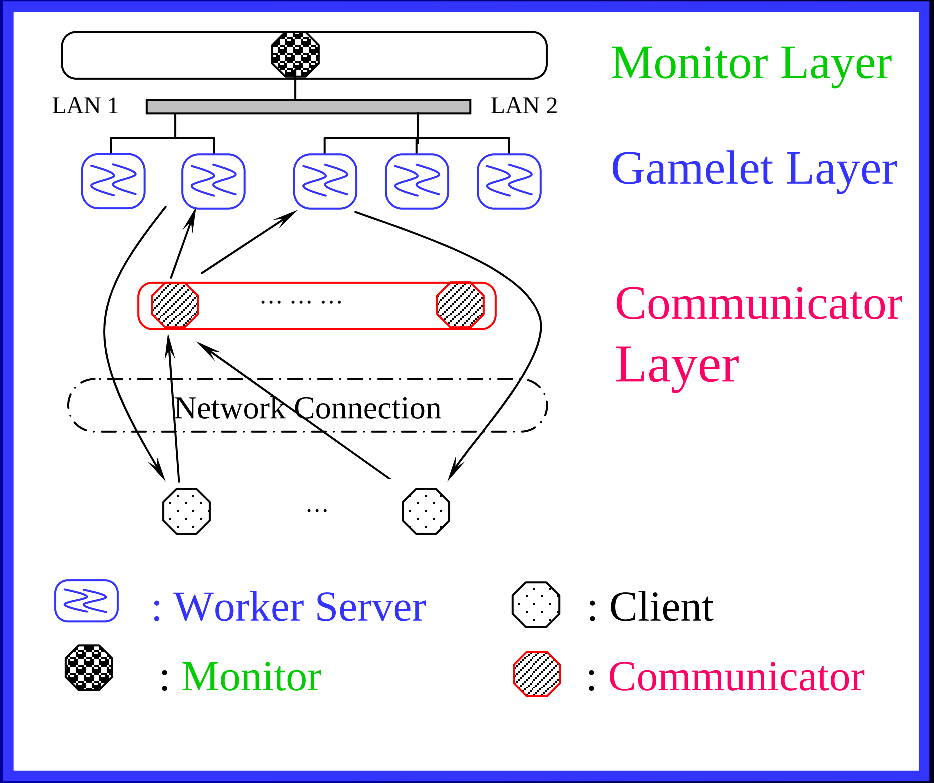
Our Approach

- Propose a service-oriented framework
  - Based on a component called gamelet

- Propose an Adaptive Gamelet Load-balancing (*AGL*) algorithm

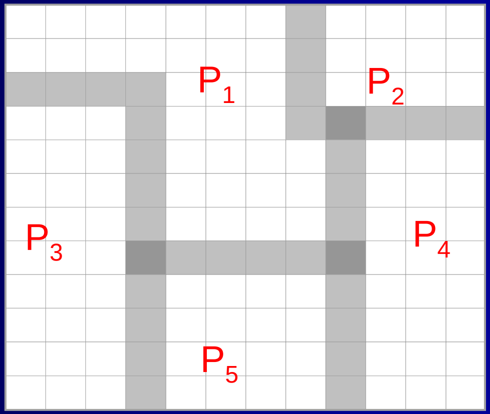
Multi-server Architecture

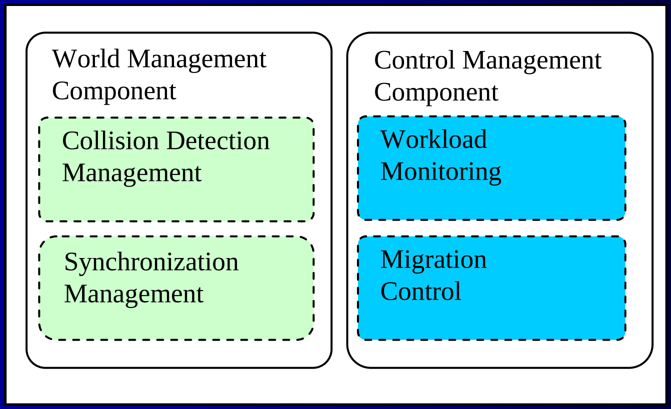
- Layered design:
  - Monitor Server
  - Worker Server
  - Communicator Server



Gamelet Concept

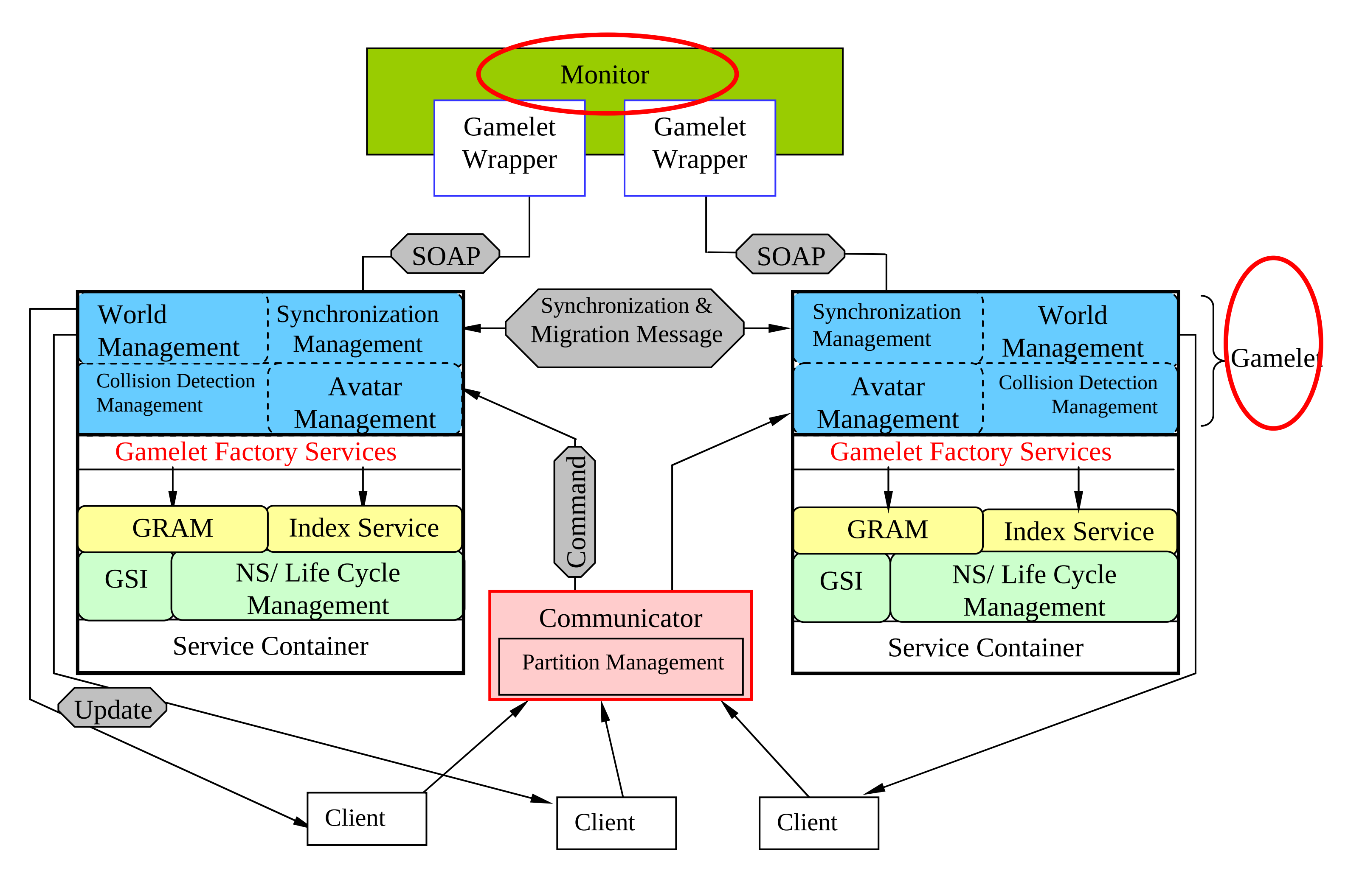
- A mobile service component that is responsible for processing the workload introduced by a partitioned virtual environment.



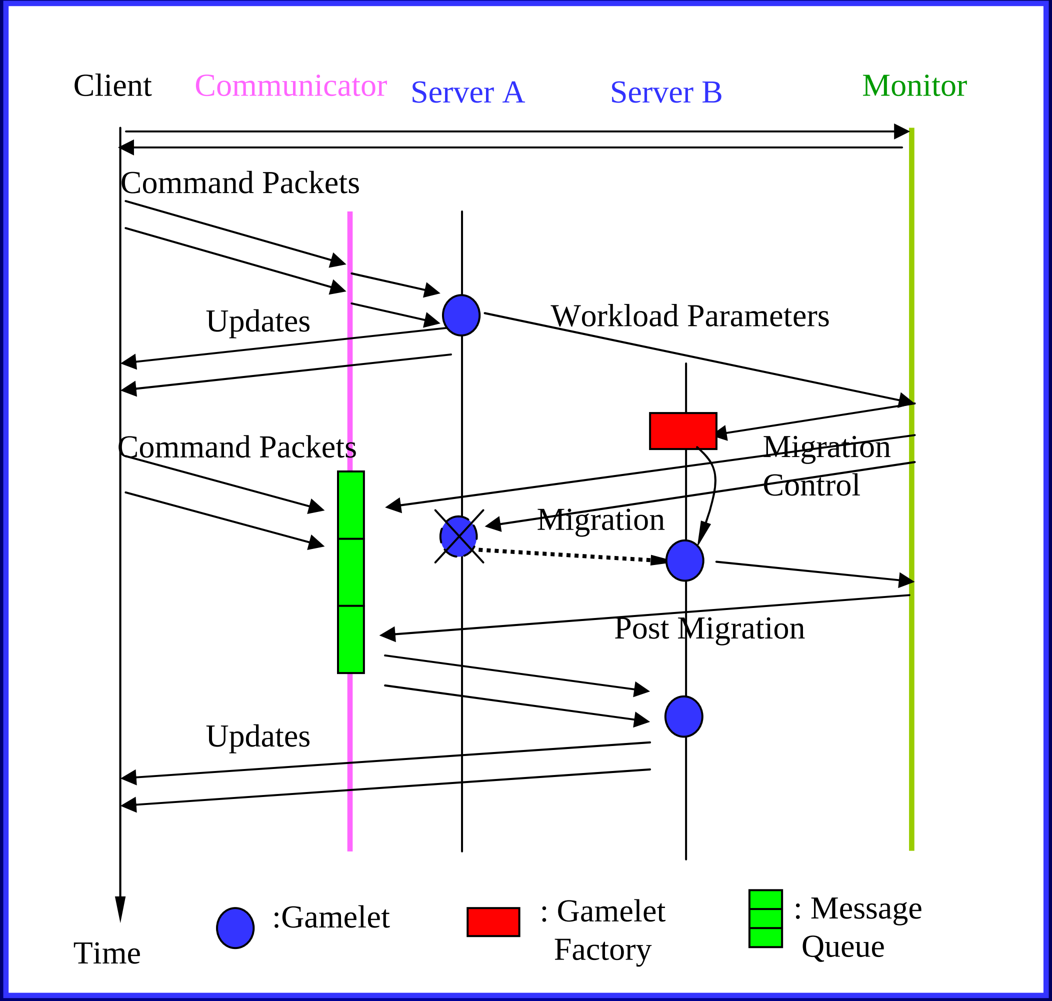


- Discussions:
  - Load awareness
  - High mobility
  - Embedded Synchronization

System Framework



Message Route/Gamelet Migration

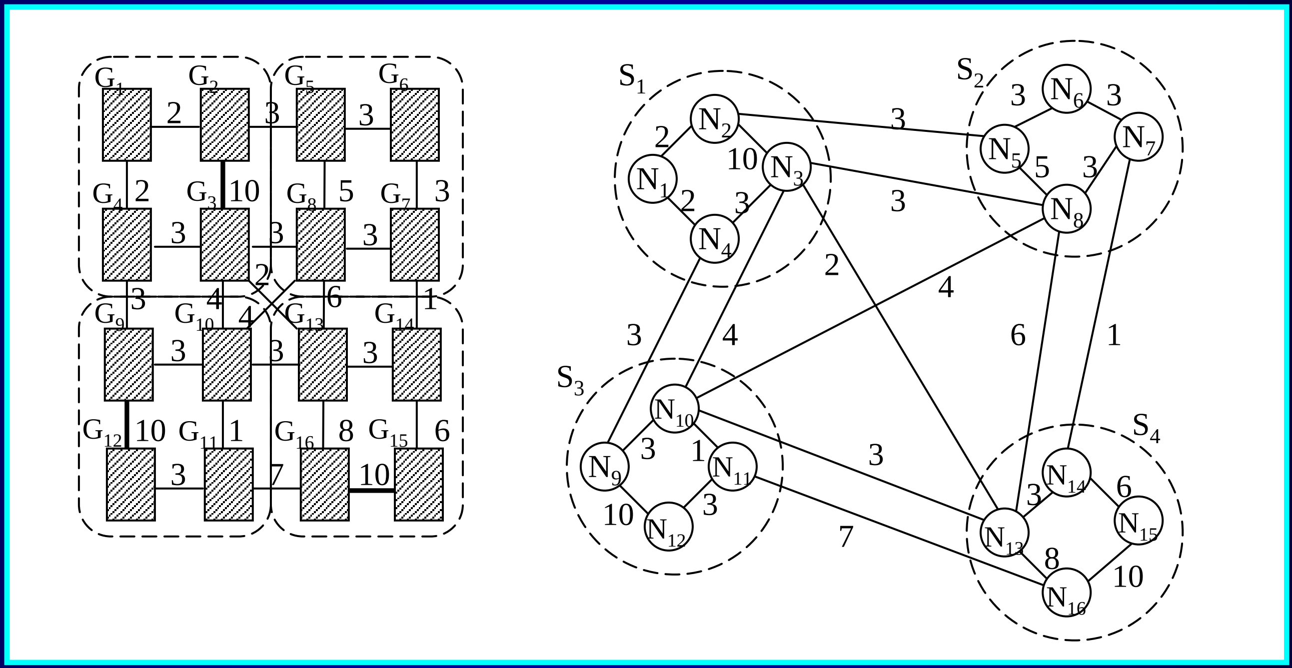


Load Balancing Strategy

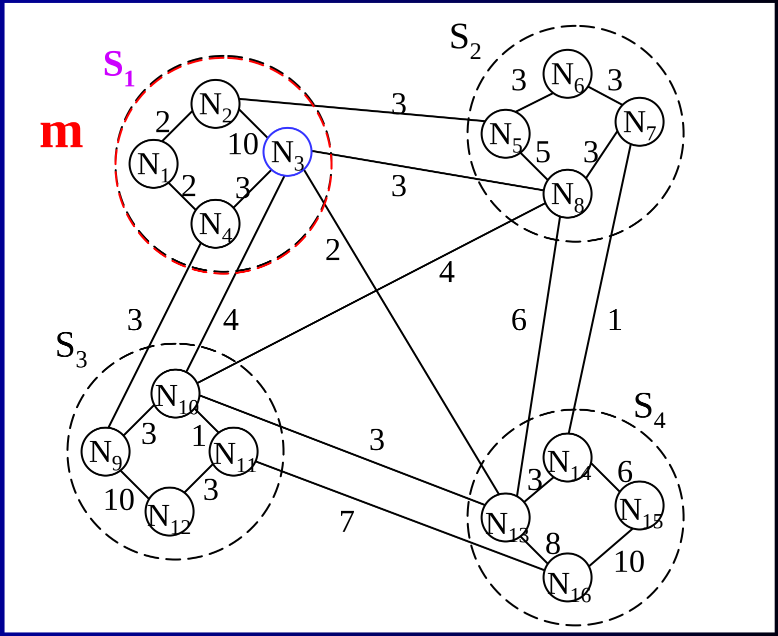
- Special characteristics of Grid environment:
  - High latency, heterogeneous machines, etc
- *Adaptive Gamelet Load-balancing* algorithm:
  - Use more accurate workload model
  - Adapt to the network latency and resource heterogeneity

AGL Algorithm

❖ Graph repartition problem



1. **For each gamelet G_i, create a node N_i in the graph G**
2. **For any two nodes N_i and N_j, if there are some inter-communications C_i,j between them, create an edge between N_i and N_j with value W_i,j = C_i,j.**



- Threshold *delta*
- *AGL* Algorithm
  1. *Select server **m, n***
  2. *Cost(G_i, **n**) = Syn'(G_i) – Syn(G_i)*
     *Syn(G_i) = Sum{ W(i,j) * Latency(m,n) }*
  3. *Select a gamelet with the smallest Cost(G_i, **n**)*
  4. *Estimate workload transferred:*
     Percentage(G_i) = Val(G_i) / Sum{ Val(G_j) }
     Val(G_i) : weighted package sending rate
  5. *Do 1-4 until original server is under threshold*
  6. *If there is still an overloaded server, add a new Grid server, go to step 1.*

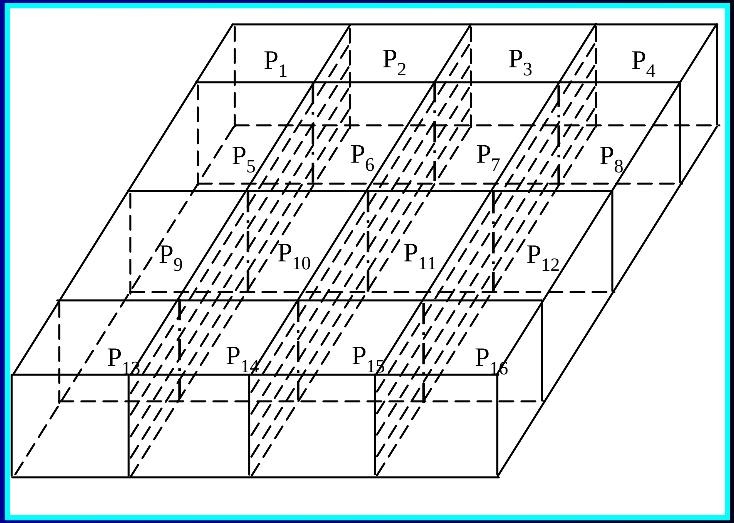
Twofold Meaning

- Adapt to the network latency
  - Cost model: synchronization cost and grid inter-server latency

- Workload evaluation based on the activities of the clients and also consider the resource heterogeneity

Prototype Design and Implementation

- Virtual environment:
  - Partitioned world: 100*100*20
  - Overlapping length: 5
- Client simulator
  - Random movement per 100ms
  - Hotspot (25ms)
  - Data packets (32 B)
- Performance parameters
  - Response time
  - System Capacity



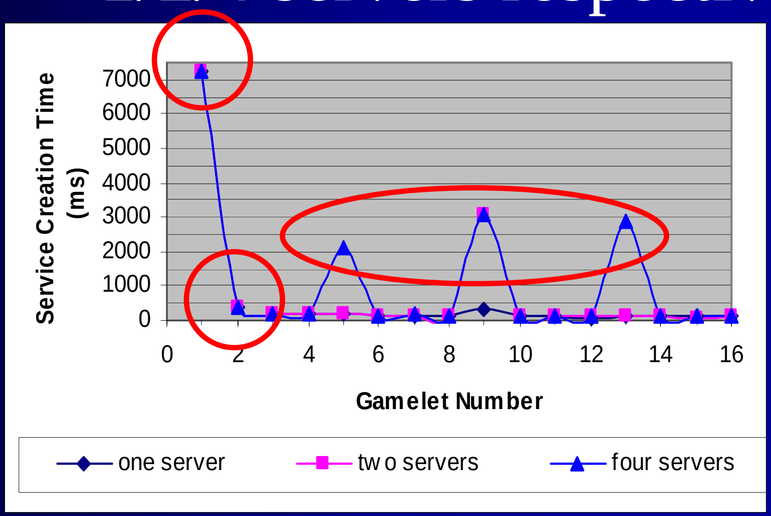
Testing Environment

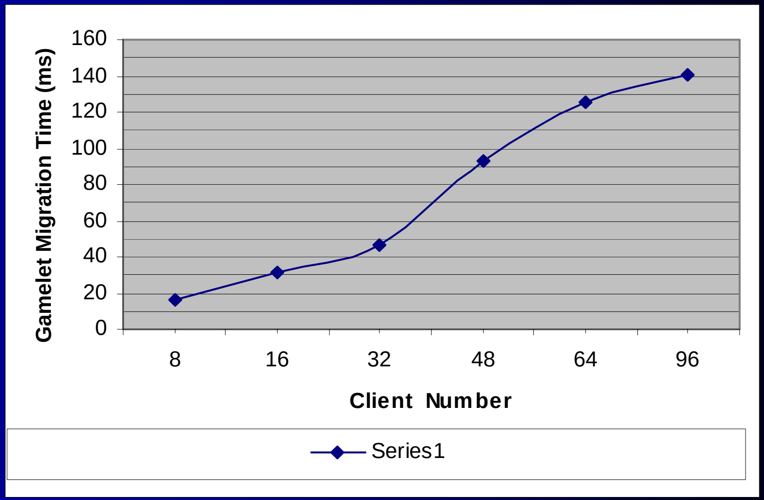
- **Gamelet and Monitor (GT3)**
- **Client simulator, communicator (J2SE 1.4.2)**
- **Server configurations**
  - Linux kernel 2.4.18, P4 2.0GHz CPU
  - 512M RAM, 100Mbps Ethernet
  - Default latency within several million seconds

Latency (ms)	S1	S2	S3	S4	S5	S6	S7	S8
S2	100	1	200	150	100	100	100	50

Gamelet Creation and Migration

- Monitor creates 16 gamelets sequentially on 1/2/4 servers respectively

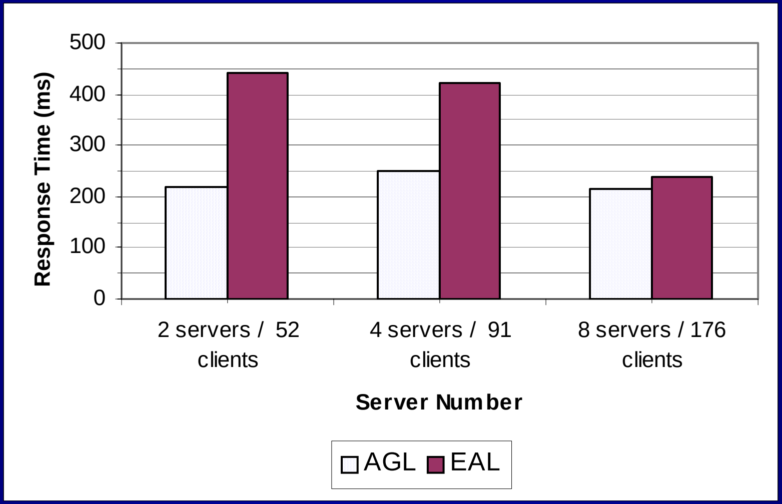




- Gamelet migration time
  - The number of avatars
  - Interaction overhead: 30 –40 ms

AGL Algorithm Evaluation

- *Delta* = 90%;
- Initially, 16 gamelets in one server; servers are added as necessary
- Compare with even-avatar algorithm (*EAL*)



Average Response Time

Figure 1

91 Clients 4 Servers	CPU Load S1	CPU Load S2	CPU Load S3	CPU Load S4	Inter-server Traffic	RT (ms) S1	RT (ms) S2	RT (ms) S3	RT (ms) S4	ART ms
AGL	90%	89%	79%	81%	276.8 Kbps	223	321	210	241	248.7
EAL	100%	100%	99%	42%	184.1 Kbps	503	572	512	101	422.0

Figure 2

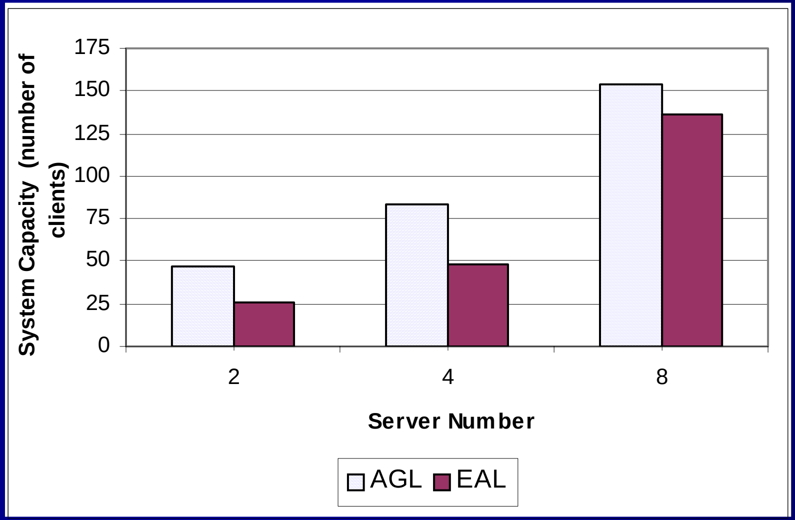
176 Clients 8 Servers	CPU Load S1	S2	S3	S4	S5	S6	S7	S8	Inter-server Traffic
AGL	90%	72%	69%	71%	71%	90%	69%	68%	891.1 Kbps
EAL	90%	88%	89%	91%	89%	90%	35%	36%	742..6 Kbps

Figure 3

176 Clients 8 Servers	RT (ms) S1	S2	S3	S4	S5	S6	S7	S8	ART (ms)
AGL	270	323	180	180	184	247	175	158	214.6
EAL	235	402	309	278	214	235	137	110	240.0

- Influence of network latency
- Influence of workload model

System Capacity



- More scalable and cost-effective
- Increase by up to 80%

Related Work

- **CittaTron** (Osaka University, 2001)
  - Multi-server networked Internet game
  - Only consider user number for load transfer
- **Cyber-walk** (CTU, 2002)
  - A distributed web walk through system
  - Partition is adjustable
  - Several hotspots -> cascading effect
- **NetEffect** (NUS, 1997)
  - VE divided into separated communities
  - Non-transparent load balancing

In our proposed gamelet-based multi-server framework, these problem are resolved.

Thank You!

Questions?